

# A BOUND FOR DICKSON'S LEMMA

JOSEF BERGER AND HELMUT SCHWICHTENBERG

**ABSTRACT.** We consider a special case of Dickson's lemma: for any two functions  $f, g$  on the natural numbers there are two numbers  $i < j$  such that both  $f$  and  $g$  weakly increase on them, i.e.,  $f_i \leq f_j$  and  $g_i \leq g_j$ . By a combinatorial argument (due to the first author) a simple bound for such  $i, j$  is constructed. The combinatorics is based on the finite pigeon hole principle and results in a descent lemma. From the descent lemma one can prove Dickson's lemma, then guess what the bound might be, and verify it by an appropriate proof. We also extract (via realizability) a bound from (a formalization of) our proof of the descent lemma.

**Keywords:** Dickson's lemma, finite pigeon hole principle, program extraction from proofs, non-computational quantifiers.

## 1. INTRODUCTION

Consider the following special case of Dickson's lemma: for any two functions  $f, g$  on the natural numbers there are two numbers  $i < j$  such that both  $f$  and  $g$  weakly increase on them, i.e.,  $f_i \leq f_j$  and  $g_i \leq g_j$ . By a combinatorial argument (due to the first author) a simple bound for such  $i, j$  is constructed. The combinatorics is based on the finite pigeon hole principle and results in a certain descent lemma. From the descent lemma one can prove Dickson's lemma, then directly guess what the bound might be, and finally verify it by an appropriate proof. We also extract (via realizability) a bound from (a formalization of) our proof of the descent lemma.

In its usual formulation, Dickson's lemma (for fixed functions) is a  $\Sigma_1^0$ -formula. In contrast, we shall prove a quantifier-free statement which implies Dickson's lemma in its usual form, but not vice versa. Our proof can be carried out in the formal system of Elementary Analysis [?, p.144], a conservative extension of Heyting arithmetic with variables and quantifiers for number-theoretic functions. In fact, we don't make use of the axiom of choice at all. Furthermore, we can restrict induction to quantifier-free formulas.

---

*Date:* March 12, 2015.

Dickson's lemma has many applications. For instance, it is used to prove termination of Buchberger's algorithm for computing Gröbner bases [?], and to prove Hilbert's basis theorem [?].

There are many other proofs of Dickson's lemma in the literature, both with and without usage of non-constructive (or "classical") arguments. The original proof of Dickson [?] and the particularly nice one by Nash-Williams [?] (using minimal bad sequences) are non-constructive, and hence do not immediately provide a bound. But it is well known that by using some logical machinery one can still read off bounds, using either Gödel's [?] Dialectica translation as in Hertz [?] or Friedman's [?]  $A$ -translation as in [?]. However, these bounds – even for the case of just two functions considered here – heavily use higher type (primitive recursive) functionals and are less perspicuous than the one obtained below.

The first constructive proof of Dickson's lemma has been given by Schütte and Simpson [?, ?], using ordinal numbers and transfinite induction up to  $\epsilon_0$ . Similar methods have been used by Sustik [?] and Martín-Mateos et al. [?]. Since initial segments of transfinite induction are used, these proofs when written in arithmetical systems require ordinary induction on quantified formulas. A different constructive proof has been given by Veldman [?]. It uses dependent choice for  $\Sigma_1$ -formulas (with parameters), and induction on  $\Pi_2$ -formulas. This proof also provides the basis of Fridlender's [?] formalization in Agda. The computational content of these proofs has not been studied; the bound involved will be very different from the present one.

## 2. A COMBINATORIAL PROOF OF DICKSON'S LEMMA

We start with a finite pigeonhole principle, in two disjunctive forms. The (rather trivial) proofs are carried out because they have computational content which will influence the term extracted from a formalization of our proofs in Section 3.

**Lemma 2.1** (FPHDisj).  $\forall_{m,f} (\exists_{i < j \leq m} f_i = f_j \vee \exists_{j \leq m} m \leq f_j)$ .

*Proof.* By induction on  $m$ . For  $m = 0$  the second alternative holds. In case  $m + 1$  let  $f_j$  be maximal among  $f_0, \dots, f_{m+1}$ . If  $m + 1 \leq f_j$  we are done. Else we have  $f_j \leq m$ . Now we apply the induction hypothesis to  $f' := f_0, \dots, f_{j-1}, f_{j+1}, \dots, f_{m+1}$ . If two of them are equal we are done. Else  $m \leq f_k$  for some  $k \neq j$  and hence  $f_j \leq f_k$ . If  $f_j = f_k$  we are done. Else we have  $f_j < f_k$ , contradicting the choice of  $j$ .  $\square$

Note that quantifier-free induction suffices here, since we only prove a property of finite lists of natural numbers.

In the key lemma 2.3 below we will need a somewhat stronger disjunctive version of the pigeonhole principle. To this end we need an injective coding  $\langle n, m \rangle$  of natural numbers which is “square-filling”, i.e. with the property

$$(1) \quad k^2 \leq \langle n, m \rangle \rightarrow k \leq n \vee k \leq m.$$

This can be achieved by

$$\begin{array}{ccccccc} & & \dots & & & & \\ & 12 & 13 & 14 & 15 & \dots & \\ & 6 & 7 & 8 & 11 & \dots & \\ & 2 & 3 & 5 & 10 & \dots & \\ & 0 & 1 & 4 & 9 & \dots & \end{array}$$

or explicitly

$$\langle n, m \rangle := \begin{cases} n^2 + m & \text{if } m < n, \\ m^2 + m + n & \text{otherwise.} \end{cases}$$

**Lemma 2.2** (FPHDisj2).

$$\forall_{f,g,k} (\exists_{i < j \leq k^2} (f_i = f_j \wedge g_i = g_j) \vee \exists_{j \leq k^2} k \leq f_j \vee \exists_{j \leq k^2} k \leq g_j).$$

*Proof.* Fix  $f, g, k$ . Use Lemma 2.1 with  $s_i := \langle f_i, g_i \rangle$  and  $m := k^2$ . In the first case from  $s_i = s_j$  we obtain  $f_i = f_j$  and  $g_i = g_j$  by the injectivity of the coding. In the second case we have some  $j \leq k^2$  with  $k^2 \leq s_j$ . From the square-filling property (1) of the coding we obtain  $k \leq f_j$  or  $k \leq g_j$ .  $\square$

As an immediate consequence we have

**Lemma 2.3** (Key).

$$\forall_{f,g,n,k} (\exists_{n < i < j \leq n+k^2+1} (f_i = f_j \wedge g_i = g_j) \vee \exists_{n < j \leq n+k^2+1} k \leq f_j \vee \exists_{n < j \leq n+k^2+1} k \leq g_j).$$

*Proof.* Use Lemma 2.2 for  $\lambda_i f_{n+1+i}$ ,  $\lambda_i g_{n+1+i}$  and  $k$ .  $\square$

Now we introduce some notation.  $\text{Mini}(f, n)$  is the first argument where  $f$  is minimal on  $\{0, \dots, n\}$ :

$$\begin{aligned} \text{Mini}(f, 0) &:= 0, \\ \text{Mini}(f, n+1) &:= \begin{cases} \text{Mini}(f, n) & \text{if } f_{\text{Mini}(f, n)} \leq f_{n+1}, \\ n+1 & \text{otherwise.} \end{cases} \end{aligned}$$

We define functions  $\Psi, \Phi, I$  and a formula  $D$  with arguments  $f, g, n$ . For readability  $f, g$  are omitted.

$$\begin{aligned}
 (2) \quad & \Psi(n) := \max\{f_{\text{Mini}(g,n)}, g_{\text{Mini}(f,n)}\}, \\
 & \Phi(n) := f_{\text{Mini}(f,n)} + g_{\text{Mini}(g,n)}, \\
 & I(n) := n + \Psi(n)^2 + 1, \\
 & D(n) := \exists_{i < j \leq n} (f_i \leq f_j \wedge g_i \leq g_j).
 \end{aligned}$$

$D(n)$  expresses that  $n$  is a bound for Dickson's lemma.

The next lemma states a crucial property of the function  $I$ : either  $I(n)$  already is a bound for Dickson's lemma, or else  $\Phi$  decreases properly when going from  $n$  to  $I(n)$ . Since this cannot happen infinitely often, iteration of  $I$  will finally give us the desired bound.

**Lemma 2.4** (Descent).  $D(I(n)) \vee \Phi(I(n)) < \Phi(n)$ .

*Proof.* Use Lemma 2.3 with  $f, g, n$  and  $\Psi(n)$ . In the first case we have  $D(I(n))$ . In the second case we have  $n < j \leq I(n)$  with  $\Psi(n) \leq f_j$ ; the third case is symmetric. Let  $i := \text{Mini}(g, n)$ . Then  $f_i \leq \Psi(n)$ . In case  $g_i \leq g_j$  we have  $D(I(n))$  and are done. Therefore assume  $g_j < g_i$ . We show (i)  $\Phi(I(n)) \leq \Phi(j)$  and (ii)  $\Phi(j) < \Phi(n)$ . From  $j \leq I(n)$  we obtain (i). For (ii) we show  $f_{\text{Mini}(f,j)} + g_{\text{Mini}(g,j)} < f_{\text{Mini}(f,n)} + g_i$ . Now  $n < j$  implies  $f_{\text{Mini}(f,j)} \leq f_{\text{Mini}(f,n)}$ , and  $g_{\text{Mini}(g,j)} \leq g_j < g_i$ .  $\square$

From Lemma 2.4 we construct a bound for Dickson's lemma. Let

$$I^0(n) := n, \quad I^{m+1}(n) := I(I^m(n)).$$

**Lemma 2.5.**  $D(I^n(0)) \vee \Phi(I^n(0)) + n \leq \Phi(0)$ .

*Proof.* Induction on  $n$ . Step  $n \mapsto n+1$ . Applying Lemma 2.4 to  $I^n(0)$  gives  $D(I^{n+1}(0)) \vee \Phi(I^{n+1}(0)) < \Phi(I^n(0))$ . In the second case we have

$$\Phi(I^{n+1}(0)) + n + 1 < \Phi(I^n(0)) + n + 1 \leq \Phi(0) + 1$$

The latter inequality follows from the induction hypothesis, since  $D(I^n(0))$  implies  $D(I^{n+1}(0))$ .  $\square$

**Proposition 2.6.**  $D(I^{f_0+g_0+1}(0))$ .

*Proof.* Apply Lemma 2.5 to  $\Phi(0) + 1$ .  $\square$

This bound is far from optimal: already for

$$f_n := \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{else} \end{cases} \quad g_n := 0$$

with optimal bound 2 we have

$$I^{f_0+g_0+1}(0) = I^2(0) = I(I(0)) > I(0) = \Psi(0)^2 + 1 = 2.$$

Can we extend this proof to show Dickson's lemma for finitely many functions? For instance for three functions a corresponding version of the key lemma holds:

$$\begin{aligned} \forall_{f,g,h,n,k} (\exists_{n < i < j \leq n+k^4+1} (f_i = f_j \wedge g_i = g_j \wedge h_i = h_j) \vee \\ \exists_{n < j \leq n+k^4+1} k \leq f_j \vee \exists_{n < j \leq n+k^4+1} k \leq g_j \vee \exists_{n < j \leq n+k^4+1} k \leq h_j) \end{aligned}$$

(Proof. Apply the original key lemma to  $\langle f, g \rangle, h, n$  and  $k^2$ ). We can also define a measure function  $\Phi(n) := f_{\text{Mini}(f,n)} + g_{\text{Mini}(g,n)} + h_{\text{Mini}(h,n)}$ . A natural candidate for  $\Psi$  is

$$\Psi(n) := \max\{f_{\text{Mini}(g,n)}, f_{\text{Mini}(h,n)}, g_{\text{Mini}(f,n)}, g_{\text{Mini}(h,n)}, h_{\text{Mini}(f,n)}, h_{\text{Mini}(g,n)}\}$$

and a natural candidate for  $I$  is  $I(n) := n + \Psi(n)^4 + 1$ . But the corresponding version of the descent lemma is false: let  $n := 2$  and

$$\begin{aligned} f &:= (0, 1, 1, 1, 0, f_5, \dots), \\ g &:= (1, 0, 1, 0, 1, g_5, \dots), \\ h &:= (1, 1, 0, 0, 0, h_5, \dots). \end{aligned}$$

Then  $\Phi(n) = 0$ ,  $\Psi(n) = 1$ ,  $I(n) = 4$ , and we neither have  $D(I(n))$  nor  $\Phi(I(n)) < \Phi(n)$ . – However, it may well be that a more refined form of the present approach works. We leave this for future research.

### 3. EXTRACTING COMPUTATIONAL CONTENT

In the following, we demonstrate how a bound for Dickson's lemma can be extracted from a proof of the existence of such a bound. The proof we will use is essentially the one presented in Section 2, i.e., it is based on the descent lemma 2.4. We will then apply the realizability interpretation to obtain the bound. In fact, the bound will be *machine* extracted from a formalization of the existence proof.

In more detail, we shall use that  $I$  is increasing (i.e.,  $n < I(n)$ ) and that from  $D(n)$  and  $n < m$  we can infer  $D(m)$ . Then we prove the existence of a bound by general induction with measure  $\Phi$ .

**3.1. General induction and recursion.** We first explain general induction w.r.t. a measure, and the corresponding definition principle of general recursion.

General induction allows recurrence to all points “strictly below” the present one. In applications it is best to make the necessary comparisons w.r.t. a measure function  $\mu$ ; for simplicity we restrict ourselves to the case where  $\mu$  has values in the natural numbers, and the ordering we refer to is the standard  $<$ -relation. The principle of general induction then is

$$\forall_{\mu,x}(\text{Prog}_x^\mu Px \rightarrow Px),$$

where  $\text{Prog}_x^\mu Px$  expresses “progressiveness” w.r.t.  $\mu$  and  $<$ , i.e.,

$$\text{Prog}_x^\mu Px := \forall_x(\forall_y(\mu y < \mu x \rightarrow Py) \rightarrow Px).$$

It is easy to see that in our special case of the  $<$ -relation we can prove general induction from structural induction. However, it will be convenient to use general induction as a primitive axiom, for then the more efficient general recursion constant  $\mathcal{F}$  will be extracted. It is defined by

$$\mathcal{F}\mu x G = Gx(\lambda_y[\text{if } \mu y < \mu x \text{ then } \mathcal{F}\mu y G \text{ else } \varepsilon]),$$

where  $\varepsilon$  denotes a canonical inhabitant of the range. It is easy to prove that  $\mathcal{F}$  is definable from an appropriate structural recursion operator.

**3.2. Non-computational quantifiers.** We now use general induction in our constructive proof of Dickson’s lemma. However, we have to be careful with the precise formulation of what we want to prove. We are not interested in the pair  $i, j$  of numbers where both  $f$  and  $g$  increase, but only in a bound telling us when at the latest this must have happened. Therefore the existential quantifiers  $\exists_{i,j}$  must be made “uniform” (i.e., non-computational); it will be disregarded in the realizability interpretation. Such non-computational quantifiers have first been introduced in [?, ?]; in [?] this concept is extended to all connectives and discussed in detail. Let

$$D'(n) := \exists_{i < j \leq n}^u (f_i \leq f_j \wedge g_i \leq g_j).$$

Using this non-computational form of  $D(n)$  we modify Lemma 2.4 to

**Lemma 3.1** (Descent<sup>nc</sup>).  $D'(I(n)) \vee \Phi(I(n)) < \Phi(n)$ .

Note that the computational content of a proof of this lemma is that of a functional mapping two unary functions and a number into a boolean. From Lemma 3.1 we obtain as before a modification of Proposition 2.6 to

**Proposition 3.2** (Bound for Dickson's lemma).

$$\forall_{f,g,n} \exists_k (I(n) \leq k \wedge D'(k)).$$

*Proof.* By general induction with measure function  $\Phi$ . We fix  $f, g$  and prove progressiveness of the remaining  $\forall_n \exists_k$ -formula. Therefore we can assume as induction hypothesis that for all  $m$  with  $\Phi(m) < \Phi(n)$  we have

$$\exists_k (I(m) \leq k \wedge D'(k)).$$

We must show

$$\exists_k (I(n) \leq k \wedge D'(k)).$$

By Lemma 3.1 we know  $D'(I(n)) \vee \Phi(I(n)) < \Phi(n)$ . In the first case we have  $D'(I(n))$  and can take  $k := I(n)$ . In the second case we apply the induction hypothesis to  $I(n)$ . It provides a  $k$  with  $I(I(n)) \leq k$  and  $D'(k)$ . But  $I(n) \leq I(I(n))$  since  $n < I(n)$ .  $\square$

**3.3. Formalization and extraction.** The formalization<sup>1</sup> (in Minlog<sup>2</sup>) of the proof above is now routine. The term extracted from it is

```
[f,g,n](GRecGuard nat nat)(Phi f g)n
([n0,f1][if (cDesc f g n0) (I f g n0) (f1(I f g n0))])
True
```

To explain this term we rewrite it in the notation above

$$\lambda_{f,g,n} \mathcal{F} \mu n G$$

with measure  $\mu$  and step function  $G$  defined by

$$\mu := \Phi,$$

$$G(n, h) := \begin{cases} I(n) & \text{if } \text{cDesc}(n), \text{ i.e., } D'(I(n)), \\ h(I(n)) & \text{otherwise, i.e., } \Phi(I(n)) < I(n), \end{cases}$$

where for readability we again omit the arguments  $f, g$  from  $\Phi, I, \text{cDesc}$ . The functions  $\Phi, I$  are defined as in (2), and  $\text{cDesc}$  is the computational content of Lemma 3.1:

```
[f,g,n][case (cKey f g n(f(Mini g n)max g(Mini f n)))
((DummyL nat ysum nat) -> True)
(Inr nn ->
[case nn
((InL nat nat)n0 ->
(cNatLeLtCases boole)(g(Mini g n))(g n0)True False)
```

<sup>1</sup>Available at [git/minlog/examples/arith/dickson.scm](https://git.minlog.org/examples/arith/dickson.scm)

<sup>2</sup>See <http://www.minlog-system.de>

```
((InR nat nat)n0 ->
  (cNatLeLtCases boole)(f(Mini f n))(f n0)True False))]]
```

Here  $nn$  is a variable of type  $\mathbf{N} + \mathbf{N}$  with  $\mathbf{N}$  the type of natural numbers, and  $cNatLeLtCases$ :

```
(Rec nat=>nat=>alpha=>alpha=>alpha)n
([n0,x,x0][case n0 (0 -> x0) (Succ n1 -> x)])
([n0,h,n1,x,x0][case n1 (0 -> x) (Succ n2 -> h n2 x x0)])
```

is the computational content of the (simple) proof of

$$\forall_{n,m}((n \leq m \rightarrow P) \rightarrow (m < n \rightarrow P) \rightarrow P)$$

expressing case distinction w.r.t.  $\leq$  and  $<$ .

$cKey$  is the computational content of Lemma 2.3:

```
[f,g,n,n0]
[case (cFPHDisjTwo([n1]f(Succ(n+n1)))([n1]g(Succ(n+n1)))n0)
  ((DummyL nat ysum nat) -> (DummyL nat ysum nat))
  (Inr nn ->
    Inr[case nn
      ((InL nat nat)n1 -> (InL nat nat)(Succ(n+n1)))
      ((InR nat nat)n1 -> (InR nat nat)(Succ(n+n1)))]]]
```

which uses  $cFPHDisjTwo$ :

```
[f,g,n][if (cFPHDisj(n*n)
  ([n0][if (g n0<f n0)
    (f n0*f n0+g n0)
    (g n0*g n0+g n0+f n0)])])
  ([ij](DummyL nat ysum nat))
  ([n0]
    Inr[if (cCodeSqFill(f n0)(g n0)n)
      ((InL nat nat)n0)
      ((InR nat nat)n0)])]
```

which in turn depends on  $cCodeSqFill$ :

```
[n,n0,n1](Rec nat=>nat=>boole)n([n2]False)
  ([n2,(nat=>boole),n3]
    [case n3 (0 -> True) (Succ n -> (nat=>boole)n)])
  n0
```

and  $cFPHDisj$ :

```
[n](Rec nat=>(nat=>nat)=>nat@@nat ysum nat)n
  ([f](InR nat nat@@nat)0)
  ([n0,d,f]
```



```

[let n1
  [if (f(Succ n0)<=f(Maxi f n0)) (Maxi f n0) (Succ n0)]
  [if (Succ n0<=f n1)
    ((InR nat nat@@nat)n1)
    [if (d([n2][if (n2<n1) (f n2) (f(Succ n2))]))
      ([ij]
        (InL nat@@nat nat)
        [if (right ij<n1)
          ij
          ([if (left ij<n1)
            (left ij)
            (Succ left ij)]@Succ right ij)]]
      ([n2]
        [if (n2<n1)
          ((cNatLeCases nat@@nat ysum nat)(f n2)(f n1)
            ((InL nat@@nat nat)(0@0))
            ((InL nat@@nat nat)(n2@n1)))
          ((cNatLeCases nat@@nat ysum nat)(f(Succ n2))(f n1)
            ((InL nat@@nat nat)(0@0))
            ((InL nat@@nat nat)(n1@Succ n2))))))]

```

To summarize, we have extracted a function which takes two functions  $f, g$  (suppressed for readability) and a number  $n$  and yields a bound. Notice that already with  $n = 0$  we obtain the desired bound for Dickson's lemma. However, the inductive argument requires the general formulation.

Our extracted bound  $B(n) := \mathcal{F}\Phi nG$  satisfies

$$\begin{aligned}
B(n) &= \mathcal{F}\Phi nG = Gn(\lambda_m[\text{if } \Phi m < \Phi n \text{ then } \mathcal{F}\Phi mG \text{ else } \varepsilon]) \\
&= \begin{cases} I(n) & \text{if } D'(I(n)), \\ B(I(n)) & \text{if } \Phi(I(n)) < I(n). \end{cases}
\end{aligned}$$

by Lemma 3.1, which also guarantees termination:  $B(n)$  will call itself at most  $I(n)$  times. As long as the iterations  $I(n), I^2(n), \dots, I^m(n)$  decrease w.r.t. the measure  $\Phi$ , the next iteration step is done. However, as soon as Lemma 3.1 goes to its “left” alternative (i.e.,  $D'(I(n))$  holds),  $I(n)$  is returned. Hence this extracted bound differs from the “guessed” one in Proposition 2.6 in that it does not iterate  $I$  a prescribed number of times ( $f_0 + g_0 + 1$  many) at 0, but stops when allowed to do so by the outcome of Lemma 3.1.